Claims 1-64 (canceled)

65. (new) A method for communicating between applications executing on separate independent partitions of a partitionable computer system, at least first and second ones of said partitions each operating under the control of a separate operating system, wherein communications between said first and second partitions is accomplished using standard network application interfaces without the need for providing an external network connection therebetween, said method comprising:

receiving a request made by a first application on said first partition to establish a network connection with a second application on said second partition and to send a message to the second application via the network connection;

establishing said network connection between said first partition and said second partition of the computer system through a memory region of the computer system shared by both said first partition and said second partition, wherein the connection emulates the requested network connection; and

sending the message to the second application via said network connection established through the shared memory region, whereby said network connection established through the shared memory region appears to the first and second applications as the requested network connection;

said establishing and said sending comprising:

creating a data structure in the shared memory region comprising a plurality of data segments;

writing, from the first partition on behalf of the first application, the message to one or more data segments, as needed, and updating an indication of the data segment containing the most recently written portion of the message.

66. (new) The method recited in claim 1, wherein the connection requested by the first application comprises a socket connection, and wherein the step of establishing a connection through the shared memory region comprises establishing a connection through the shared memory region that emulates a socket connection.

67. (new) The method recited in claim 65, further comprising:

reading, from the second partition on behalf of the second application, the message from said one or more data segments and updating an indication of which data segments have been read from the data structure; and

providing the message read from the data structure to the second application in accordance with an API associated with the requested network connection.

68. (new) The method of claim 65, wherein said plurality of data segments form a circular buffer, and wherein updating an indication of the data segment containing the most recently written portion of the message comprises incrementing a head index.

69. (new) The method of claim 68, wherein updating an indication of which data segments have been read from the data structure comprises incrementing a tail index.

70. (new) The method of claim 69, further comprising polling, by the receiving partition, the shared memory region to determine if the message has been written to the shared memory region.

71. (new) The method of claim 70, further comprising receiving, by the receiving partition, an interrupt initiated by the sending partition and indicating that the message has been written to the shared memory region.

72. (new) The method of claim 66, wherein the step of establishing a connection between the first partition and the second partition of the computer system that emulates a socket connection further comprises performing the following steps on the second partition:

(a) creating a socket on behalf of the second application to listen for attempts to connect thereto;

(b) receiving a connect message from the first partition that identifies a memory location of the shared memory region at which the first partition has allocated a first data area to serve as a buffer for transferring data from the first partition to the second partition;

(c) matching the received connect message to the listening socket created in step (a);

(d) allocating a second data area in the shared memory region to serve as a buffer for transferring data from the second partition to the first partition;

(e) mapping both the first and second data areas into a process space of the listening socket;

(f) initializing the second data area; and

(g) returning a connected indication to the first partition and informing the application on the second partition that the socket connection has been established.

73. (new) The method of claim 72, further comprising performing the following steps on the first partition:

(a') receiving the request from the first application to establish the socket connection with the second application.

(b') creating a connecting socket;

(c') allocating the first data area in the shared memory region;

(d') sending the connect message to the second partition that identifies the memory location of the shared memory region at which the first data area has been allocated; and

(e') upon receipt of the connected indication from the second partition, mapping the first and second data areas into a process space of the connecting socket to establish the socket connection between the first and second partitions.

74. (new) The method of claim 65, further comprising:

creating, the shared memory region, a plurality of output queues, one for each of said first second partitions, the output queue for a given partition indicating whether that partition has placed in the shared memory region a message intended for any of the other partitions and if so, identifying a buffer containing the message, each partition poling the output queues of the other partitions to determine whether those other partitions have placed any messages intended for it in the shared memory region;

receiving at the first partition from the first application, said request to send a message to the second application via the requested type of network connection;

writing, in response to the received request, the message to an available buffer in the shared memory region and indicating in the output queue of the first partition that the message has been written thereto;

determining, at the second partition, from the output queue of the first partition, that the message has been placed in the available buffer and retrieving the message from the available buffer; and

providing the message read from the data structure to the second application in accordance with an API associated with the requested network connection.

75. A computer readable medium having program code store thereon for communicating between applications executing on separate independent partitions of a partitionable computer system, at least first and second ones of said partitions each operating under the control of a separate operating system, wherein communications between said first and second partitions is accomplished using standard network application interfaces without the need for providing an external network connection therebetween, and wherein said program code, when executed, causing performance of the following:

receiving a request made by a first application on said first partition to establish a network connection with a second application on said second partition and to send a message to the second application via the network connection;

establishing a connection between the first partition and the second partition of the computer system through a memory region of the computer system shared by both the first partition and the second partition, wherein the connection emulates the requested network connection; and

sending the message to the second application via said network connection established through the shared memory region, whereby said network connection established through the shared memory region appears to the first and second applications as the requested network connection;

creating a data structure in the shared memory region comprising a plurality of data segments; and

writing, from the first partition on behalf of the first application, the message to one or more data segments, as needed, and updating an indication of the data segment containing the most recently written portion of the message.

76. The computer readable medium recited in claim 75, wherein the connection requested by the first application comprises a socket connection, and wherein the step of

establishing a connection through the shared memory region comprises establishing a connection through the shared memory region that emulates a socket connection.

77. (new) The computer readable medium recited in claim 75, wherein the program code, when executed further causes the performance of the following:

reading, from the second partition on behalf of the second application, the message from said one or more data segments and updating an indication of which data segments have been read from the data structure; and

providing the message read from the data structure to the second application in accordance with an API associated with the requested network connection.

78. (new) The computer readable medium of claim 75, wherein said plurality of data segments form a circular buffer, and wherein updating an indication of the data segment containing the most recently written portion of the message comprises incrementing a head index.

79. (new) The computer readable medium of claim 78, wherein updating an indication of which data segments have been read from the data structure comprises incrementing a tail index.

80. (new) The computer readable medium of claim 75, wherein the program code, when executed, further causes the processor to poll, by the receiving partition, the shared memory region to determine if the message has been written to the shared memory region.

81. (new) The computer readable medium of claim 70, wherein the program code, when executed, further causes the processor to receive, by the receiving partition, an interrupt initiated by the sending partition and indicating that the message has been written to the shared memory region.

82. (new) The method of claim 76, wherein the establishing a connection between the first partition and the second partition of the computer system that emulates a socket connection further comprises performing the following steps on the second partition:

(a) creating a socket on behalf of the second application to listen for attempts to connect thereto;

(b) receiving a connect message from the first partition that identifies a memory location of the shared memory region at which the first partition has allocated a first data area to serve as a buffer for transferring data from the first partition to the second partition;

(c) matching the received connect message to the listening socket created in step (a);

(d) allocating a second data area in the shared memory region to serve as a buffer for transferring data from the second partition to the first partition;

(e) mapping both the first and second data areas into a process space of the listening socket;

(f) initializing the second data area; and

(g) returning a connected indication to the first partition and informing the application on the second partition that the socket connection has been established.

83. (new) The method of claim 82, wherein the step of establishing a connection further comprises performing the following steps on the first partition:

(a') receiving the request from the first application to establish the socket connection with the second application;

(b') creating a connection socket;

(c') allocating the first data area in the shared memory region;

(d') sending the connect message to the second partition that identifies the memory location of the shared memory region at which the first data area has been allocated; and

(e') upon receipt of the "connected" indication from the second partition, mapping the first and second data areas into a process space of the connecting socket to establish the socket connection between the first and second partitions.

84. (new) The computer readable medium of claim 75, wherein the program code, when executed, further causes performance of the following:

creating, in the shared memory region, a plurality of output queues, one for each of said first and second partitions, the output queue for a given partition indicating whether that partition has placed in the shared memory region a message intended for any of the

other partitions and if so, identifying a buffer containing the message, each partition polling the output queues of the other partitions to determine whether those other partitions have placed any messages intended for it in the shared memory region;

receiving at the first partition from the first application, said request to send a message to the second application via the requested type of network connection;

writing, in response to the received request, the message to an available buffer in the shared memory region and indicating in the output queue of the first partition that the message has been written thereto;

determining, at the second partition, from the output queue of the first partition, that the message has been placed in the buffer and retrieving the message from the buffer; and

providing the message read from the data structure to the second application in accordance with an API associated with the requested network connection.

85. (new) A computer system comprising:

a plurality of processing modules, groups of one or more processing modules being configured as separate independent partitions within the computer system, each partition operating under the control of a separate operating system, wherein communications between said first and second partitions is accomplished using standard network application interfaces without the need for providing an external network connection there between;

a main memory to which each processing module is connected, the main memory having defined therein at least one shared memory region to which at least said first and second ones of said partitions have shared access; and

program code, executing on each of at least said first partition and said second partition of the computer system, said program code establishing a connection between a first application on said first partition and a second application on said second partition through the shared memory region, wherein the connection through the shared memory region emulates a network connection requested by one of the applications;

wherein said program code executing on each of said first and second partitions comprises a shared memory service provider that serves as an interface between a component of the computer system that provides an API through which said first

application can make said request for a network connection and the shared memory region of the main memory through which the emulated network connection is established;

wherein the shared memory service provider on each of said first and second partitions establishes a data structure in the shared memory region through which data is transferred from that partition to the shared memory service provider on the other partition.

wherein the data structure comprises:

a plurality of data segments, each of the plurality of data segments for storing network message data to be sent from a sending shared memory service provider to a receiving shared memory service provider;

a control segment for controlling reading and writing of data in the plurality of data segments, the control segment comprising:

a first portion comprising:

a first field for storing an indication of the data segment containing the most recently written network message data; and

a second field for storing an indication of the data segment containing the earliest written, but ot read, network message data;

and

a plurality of second portions, each second portion corresponding to one of the plurality of data segments for control of the data segment, each second portion comprising:

a first field for storing an indication of the beginning of network message data within the data segment; and

a second field for storing an indication of the end of network message data within the data segment.

86. (new) The computer system recited in claim 85, wherein the first portion further comprises:

a third field for storing an indication that the sending shared memory service provider is waiting to send the network message; and

a fourth field for storing an indication that the receiving shared memory service provider is waiting to receive the network message.

87. (new) The computer system recited in claim 85, wherein each second portion further comprises a third field for storing an indication of a length of network message data within the data segment.

88. The computer system recited in claim 87, wherein the plurality of data segments are linked to form a circular buffer, and wherein each second portion further comprises:

a fourth field for storing an indication of the next data segment in the circular buffer, and

a fifth field for storing an indication that the data segments contains a last portion of a network message stored across a plurality of data segments.

89. (new) The computer system recited in claim 85, wherein the computer system provides a resource through which the shared memory service provider can establish the data structure and control the transfer of data through it, the resource providing the ability to perform at least one of the following operations on the shared memory region: (i) allocate an area of the shared memory region; (ii) map and unmap an area of the shared memory region; deallocate an area of the shared memory region; (iii) send and receive signals to and from other partitions via the shared memory region; and (iv) receive status information about the shared memory region and about selected partitions.

90. (new) The computer system recited in claim 85, wherein the shared memory service provider comprises:

a dynamic link library (DLL) that executes in a user mode of the operating system of its respective partition, there being an instance of the shared memory service provider DLL in a process space of each application in the partition that may request the establishment of a network connection; and

a device driver that executes in a kernel mode of the operating system of the respective partition, there being only one instance of the device driver in each partition.

91. (new) The computer system recited in claim 85, wherein the connection established through the shared memory region emulates a socket connection.

92. (new) The computer system recited in claim 91, wherein said program code executing on each of said at least first and second partitions comprises a shared memory service provider that serves as an interface between a component of the computer system that provides an API through which an application can make a request for a socket connection and the shared memory region of the main memory through which the emulated socket connection is established.

93. (new) The computer system recited in claim 92, wherein the operating system in each partition comprises a MICROSOFT WINDOWS operating system, and wherein the component of the computer system that provides the API of the requested socket connection comprises a Winsock DLL and a Winsock Switch, the Winsock DLL forwarding a request for a socket connection made by an application in a given partition to the Winsock Switch, which Winsock Switch allows multiple service providers, each of which provide TCP/IP services, to service such a request, and wherein the shared memory service provider acts as a TCP/IP service provider so that a request from an application for a socket connection can be serviced by the shared memory service provider.

94. (new) The computer system recited in claim 93, wherein the shared memory service provider on a first partition that represents the listening side of a requested socket connection performs the following steps:

(a) creating a socket on behalf of a first application executing in the first partition in order to listen for attempts to connect thereto;

(b) receiving a connect message from the shared memory service provider on the second partition that identifies a memory location of the shared memory region at which the shared memory service provider on the second partition has allocated a first data area to serve as a buffer for transferring data from the second partition to the shared memory service provider on the first partition;

(c) matching the received connect message to the listening socket created in step (a);

(d) allocating a second data area in the shared memory region to serve as a buffer for transferring data from the first partition to the shared memory service provider on the second partition;

(e) mapping both the first and second data areas into a process space of the listening socket;

(f) initializing the second data area; and

(h) returning a "connected" indication to the shared memory service provider on the second partition and informing the application on the first partition that a socket connection has been established.

95. The computer system recited in claim 94, wherein the shared memory service provider on the second partition performs the following steps:

(a') receiving a request from an application on the second partition to establish a socket connection with the first application on the first partition;

(b') creating a connection socket on the second partition;

(c') allocating the first data area in the shared memory region;

(d') sending the connect message to the first partition that identifies the memory location of the shared memory region at which the first data area has been allocated; and

(e') upon receipt of the "connected" indication from the first partition, mapping the first and second data areas into a process space of the connecting socket to establish the socket connection between the first and second partitions.

96. (new) The computer system recited in claim 85, wherein the program code implements a polling process by which each partition polls an area within the shared memory region to determine whether any communications intended for it have been placed in the shared memory region by another partition.

97. (new) The computer system recited in claim 96, wherein the area comprises a plurality of output queues, one for each partition, the output queue for a given partition indicating whether that partition has placed in the shared memory region any communications intended for any of the other partitions, each partition polling the output queues of the other partitions to determine whether those other partitions have placed any communications intended for it in the shared memory region.

98. (new) The computer system recited in claim 97, wherein for any communications placed in the shared memory region by a sending partition and intended to be received by

another partition, the output queue of the sending partition specifies the location within the shared memory region of a buffer containing that communication.

99. The computer system recited in claim 98, wherein the program code executing on each of said first and second partitions further comprises a shared memory driver that receives a request to send a message to an application on another partition, the request having been made in accordance with the application programming interface (API) associated with the requested type of network connection, and that, in response to the request, causes the message to be placed in an available buffer in the shared memory region and causes an indication of the message to be placed in the output queue of the sending partition.

100. (new) The computer system recited in claim 99, wherein the shared memory driver on each partition implements a same interface as a network device driver to enable application programs and the operating system on that partition to send communications to other partitions via the shared memory region in the same manner that communications are sent to other computer systems over a network via a network interface card.